Fusion Engineering and Design 87 (2012) 2136–2139

Contents lists available at SciVerse ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

Visualization of target inspection data at the National Ignition Facility

Daniel Potter*, Nick Antipa

Lawrence Livermore National Laboratory, United States

ARTICLE INFO

Article history:
Available online 4 July 2012

Keywords:
NIF
Visualization
WebGL
Web application

ABSTRACT

As the National Ignition Facility continues its campaign to achieve ignition, new methods and tools will be required to measure the quality of the target capsules used to achieve this goal. Techniques have been developed to measure capsule surface features using a phase-shifting diffraction interferometer and Leica Microsystems confocal microscope. These instruments produce multi-gigabyte datasets which consist of tens to hundreds of files. Existing software can handle viewing a small subset of an entire dataset, but none can view a dataset in its entirety. Additionally, without an established mode of transport that keeps the target capsules properly aligned throughout the assembly process, a means of aligning the two dataset coordinate systems is needed. The goal of this project is to develop web based software utilizing WebGL which will provide high level overview visualization of an entire dataset, with the capability to retrieve finer details on demand, in addition to facilitating alignment of multiple datasets with one another based on common features that have been visually identified by users of the system.

Published by Elsevier B.V.

1. Introduction

The National Ignition Facility (NIF) [1] at Lawrence Livermore National Laboratory is the world's largest laser. It has been built with the goal of being the first facility to demonstrate controlled laser-driven ignition. This is achieved through a process called inertial confinement fusion (ICF), whereby extreme force is applied to a hollow spherical, BB-sized capsule containing fusion fuel, causing the fuel to compress and heat until a fusion reaction occurs. Due to the extreme temperature and pressure required to achieve ignition, energy must be distributed very uniformly around the capsule during implosion. Isolated features on the surface of the capsule can cause an uneven implosion, leading to radiative cooling of the implosion core and lowering the probability of achieving ignition.

To better evaluate the target manufacturing process, and to gain insight on how the shape of capsule surfaces affect the performance of NIF experiments, each spherical target capsule's surface is measured twice: first using a phase-shifting diffraction interferometer (PSDI) [2] then, later in the assembly process, a Leica Microsystems "Leica DCM-3D", a surface profiling programmable array confocal microscope. These instruments each produce a set of images, containing both topography and reflectivity information, that cover the entire capsule surface. A single dataset can produce up to 1000 images, totaling several gigabytes in size. These images can be viewed individually, but a method of viewing the complete 3D dataset in its spherical geometry is desired.

This paper introduces an image based visualization system for data exploration of target shells at the NIF. The visualization software combines multiple image sets into a single visualization in order to facilitate alignment of data sets, and to provide a method of navigating the data in ways that are not possible with existing tools.

The next section describes the design of the overall system, from the acquisition and storage of this data, to its visualization.

2. System design

Software has been developed to visualize this data in a three dimensional space, much like Google Earth can be used to view geographical data. Fig. 1 shows the system architecture of the application.

2.1. Inspection and data storage

Raw image data and metadata are collected from the microscopes and uploaded to an Oracle database. For the Leica microscope data, an image analysis program is run which identifies features of interest in the individual images and records their locations. During the imaging process for both systems, the capsule is mounted to a rotating stage which allows the entire surface to be imaged. The theta/phi position of the rotating stage at the time an image is taken is stored as metadata with every image,

* Corresponding author.
E-mail addresses: potter15@llnl.gov (D. Potter), antipa1@llnl.gov (N. Antipa).

0920-3796/$ – see front matter. Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.fusengdes.2012.04.019

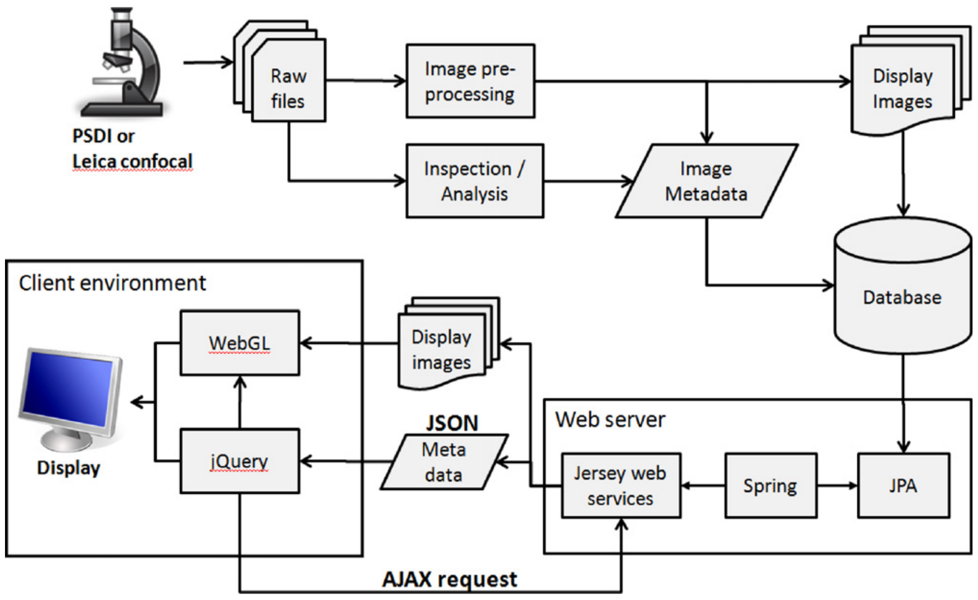

Fig. 1. Raw microscope image data is picked up by image processing routines which analyze and pre-process the images for visualization. The visualization layer is implemented as a web application.

as well as other important metadata such as pixel and image size. Display images are created from the raw image data during the upload phase. Images are stored at multiple resolutions so that lower resolution images can be downloaded first in the visualization, and higher resolution images can be downloaded on demand. This reduces the initial wait time for the user. These display images are stored in the PNG image format, and add less than 10% to the storage requirements of the raw data.

2.2. *Web application architecture*

Our web application relies on several open source technologies and frameworks. On the database and server side framework end we use JPA and Spring. JPA, or the Java Persistence API, is a framework which allows us to easily map database objects to java classes, simplifying communication with the database. We use the Spring framework to create and inject these data access objects into the application during initialization of the application server.

On the client side we build our user interface using jQuery, and communicate with the server side Java code and JPA data objects using RESTful web services. jQuery is a library built on top of Javascript which simplifies development of Javascript code and solves many cross browser compatibility problems associated with Javascript. We have found that jQuery facilitates a separation of concerns with different modules of the UI with its simplified event model and extended libraries such as JavascriptMVC, which add some object oriented capabilities to javascript programs. jQuery also allows us to easily make AJAX (i.e. XMLHttpRequests) to provide a rich internet application with quick response times.

Our AJAX requests are made against RESTful web services, web services made accessible over a URL using the HTTP protocol. These serve as a data source that can be directly and easily accessed through jQuery's AJAX interface. We use the Java based Jersey API for our web services. We find that a simple service can be set up in a minimal amount of time using this framework. The Jersey API also has plug-ins that can automatically convert java objects into JSON data formats. JSON, or JavaScript Object Notation, is a format similar to XML but tends to be more concise. It is also a convenient format to use in Javascript based applications.

Converting our database model objects into JSON enables us to use them in jQuery/Javascript in the same way we would use them in server side java code. This simplifies the design of the system as the representation of objects is consistent at all layers of the application. 3D rendering is achieved using the WebGL framework, while all other UI elements and logic is handled using jQuery based libraries.

2.3. *Functionality of the application*

The functionality of the application is inspired by Google Earth, where users can rotate, zoom, and pick out sites of interest on an interactive 3d visualization of the Earth. Each image is positioned in a 3D space, creating a spherical surface that the user can interactively rotate and zoom in on.

To position images, it first queries the radius of the target, the theta/phi coordinates of the stage relative to the microscope and pixel width/height of each image belonging to the data set. A geometry patch is generated representing the surface curvature spanning the image, and each image is texture mapped onto its corresponding geometry patch. Simple quads were originally used, but the curved geometry makes some overlapping image features line up better, and makes the overall target look more spherical when completely zoomed out. When the image is first loaded, its spherical coordinate values are converted to Cartesian coordinate space, this is where the geometry is placed during rendering. Before rendering occurs, each geometry patch is rotated about its local Z axis by the images phi value, and rotated about its local X axis by the images theta value, making it face outwards at the proper angle. The final result is displayed in Fig. 2.

Fig. 2 should give an idea of the utility of this visualization due to the fact that over 200 images are involved. The darker borders of the images in the smaller grayscale dataset make their size obvious. Images in the other dataset are roughly the same size. It would clearly be much harder to view these datasets image by image, or even in groups of images. An advantage to being able to view the entire dataset at once is that it gives users a spatial context for each individual image.

Another feature provided is the listing of features of interest found during the inspection phase mentioned in Section 3.1. Users are able to sort this list by any attribute of interest. Upon clicking a row in the list, the view automatically rotates to the location of the feature that was clicked. This is done by calculating the angle